Citation

European Design Automation Conference >archive
Proceedings of the conference on European design automation >toc
1992 , Congress Centrum Hamburg, Hamburg, Germany

On modeling integrated design environments

Consider this for next time.

Authors
Christoph Hübel
Detlev Ruland
Ernst Siepmann

Sponsors
IEEE-CS : Computer Society
SIGDA : ACM Special Interest Group on Design Automation

Publisher
IEEE Computer Society Press Los Alamitos, CA, USA

Pages: 452 - 458 Series-Proceeding-Article
Year of Publication: 1992
ISBN:0-8186-2780-8

> full text > references > cilings > index terms > peer to peer

> Discuss > Similar > Review this Article Save to Binder

> BibTex Format

FULL TEXT: Access Rules

pdf 770 KB

REFERENCES

Note: OCR errors may be found in this Reference List extracted from the full text article. ACM has opted to expose the complete List rather than only correct and linked references.

ARF91 Wayne Allen , Douglas Rosenthal , Kenneth Fiduk, The MCC CAD framework methodology management system, Proceedings of the 28th conference on ACM/IEEE design automation conference, p.694-698, June 17-22, 1991, San Francisco, California, United States

BD86 Michael L. Bushnell , S. W. Director, VLSI CAD tool integration using the Ulysses environment, Proceedings of the 23rd ACM/IEEE conference on Design automation, p.55-61, July

On Modeling Integrated Design Environments

Christoph Hübel, Detlev Ruland, Ernst Siepmann

University of Kaiserslautern, Siemens AG, Siemens AG
D-6750 Kaiserslautern, D-8000 München, D-8000 München

Abstract

Most existing approaches towards modeling design methodologies concentrate on modeling of design flows and design tools. Only a few of them consider the relationships between design methodology and design object modeling. Our approach towards modeling the heterogeneous aspects of design environments is based on a paradigm of **separation** *and* **integration** *yielding an adequate, well structured, non-redundant, and integrated* **design model** *for generic design environments. The design model consists of five* **partial models**: *(1) design flow model, (2) design tool model, (3) design structure model, (4) design object model, and (5) design subject model. The* **design structure model** *is introduced for modeling on a higher level of abstraction exactly those aspects of design objects which are necessary for design methodology management. The applied concept is called* **design object abstraction**.

1 Introduction

Computer Systems for supporting engineers in designing complex technical objects are topically of great interest in design automation community. (Remark: In this paper, we chose an example from the field of electronics and VLSI design, but we suggest that the idea standing behind has a rather general nature and would additionally be valuable in other fields of design automation, e.g. in the area of mechanical engineering).

The term **CAD framework** is commonly used to address a system or a system architecture that promise to be a suitable, i.e. extensible, adaptable, as well as homogeneous environment dealing with design processes, tools, and objects. CAD frameworks commonly provide the following basic services (cf. [HNSB90]): data management, data representation (including versioning), user interface, design methodology management, and some low level services (e.g. physical data management, process management, etc.).

Design Methodology Management is classified as a meta service, because it is a rather abstract service supervising the other services. But, it is not (or only partially) considered in most of today's CAD framework development approaches. Recently, CFIs Design Methodology Management Technical Subcommittee (DMM-TSC) presented a perspective towards Design Methodology Management (cf. [FKKP90]). A sequence of hierarchically structured tasks spanning all design phases of the whole product life cycle. A more abstract point of view, the CFI's DMM-TSC's perspective focuses on issues of the design flow. That is, it addresses the description and the execution of design tools.

Hence, tool information are required which are relevant for design methodology management. The applied concept is called **tool abstraction**. That means, it is abstracted as much as possible from the individual properties of design tools resulting in a generic tool description which, in turn, may be parameterized. Thus, design tools can be characterized by sets of parameters. Having made known these tool parameters to a particular framework, this can successfully invoke design tools' execution (cf. [TAS90]). Beside such abstract tool information, design methodology management requires some information about design objects. For instance, it is necessary to know, whether the execution of a design tool (in order to perform a particular design task) generates a new version of a design object or an alternative. Furthermore, hierarchy information about the design objects are required for controlling the design flow and executing design tools. Today, this information is usually included in product models, e.g. EDIF (cf. e.g. [EDIF87]), STEP (cf. e.g. [STEP88]), VHDL (cf. e.g. [VHDL85]), etc. Product models describe design data in a granularity and on an abstraction level suitable for design tools. But, for design methodology management, a more coarse granularity and a higher level of abstraction is necessary. Therefore, a concept of **design data abstraction** is required that may be compared with the idea of tool abstraction.

To gain a high degree of flexibilty and a wide range of applications for CAD frameworks, interchangeability of tools, products models, and methodologies must be assured similiary. Therefore, an explicit description in an appropriate granularity and on an appropriate abstraction level of all these components is absolutely necessary.

We propose a modeling approach that considers all the different and heterogeneous aspects of design environments throughout. Our approach is based on the paradigm of **separation** and **integration**, which is well known in the database community where it is applied to get centralized information structures that show a high degree of orthogonality. Exploiting this concept for modeling design environment yields an adequate, well structured, non-redundant, and inte-

grated **design model**. The proposed design model is structured in five **partial models**: (1) design flow model, (2) design tool model, (3) design structure model, (4) design object model, and (5) design subject model. The **design structure model** is introduced for modeling exactly those aspects of design objects which are relevant for design methodology management. The benefits of the proposed approach are, that it allows us to make explicit the prevailing relationships among design tools, design flows, and structure of design objects and to describe design methodologies from an rather integrative point of view.

In section 2 the term design environment as it is used in this paper and the role of CAD frameworks are addressed. In 2 the different classes of information for representing design environments are identified. The remainder of this section focus on design methodology management. In section 3 the proposed approach is compared with other design methodolgy management modeling approaches. Finally, section 4 summerizes the main ideas and sketch out our future plans.

2 Design Model

In our view, a **design environment** consists of a design system and a CAD framework; cf. figure 1. A **design system** (CAD system) consists of **design tools** (e.g. floorplanner, router) and **design data**. The process within a design system is called **design process** [1]. The architecture of design environments shown in figure 1 represents the **design centered approach**, cf. e.g. [HNSB90]. That means, all design tools are controlled by a central agency.

Modeling and formalizing all relevant issues of real design environments marks the first and the most important step towards a system integrated support of design processes. An overall **design model** forms the basis of a CAD framework. Only the explicit description and management of all relevant aspects of design environments, design processes can be performed completely in a computer aided way by designers' interaction with design tools and CAD frameworks, respectively.

First of all, a clear semantics of the terms *model* and *modeling* is given. For instance, the terms *system model*, *data model*, and *product model* are used with different semantics. A *system model* describes the behaviour of a system. A *data model* provides description languages for modeling database schemes. A *product model* is a prototype or schema of all data, which are relevant for designing a product of a certain product class.

In this paper, the term *model* is used in the latter sense. In figure 2, the relationships among data, models, and modeling languages are explained. The **data** describe a specific object, e.g. a product or a system. Abstracting **models** are type or schema descriptions of the data on a higher level of abstraction. On the most abstract level, **modeling languages** provide tools for specifying models. Modeling languages can be also called **application model building set**, which can be used for building models or schemas, respectively.

[1] The applied term *system* stems from system dynamics theory for information processing (cf. e.g. [We89].

2.1 Information Classes in Design Environments and Their Modeling

Refering to figure 1, within design environments the following **relevant components** can be identified:

- Design data of design objects (e.g. VLSI cells)
- Design tools (e.g. floorplanners, simulators)
- Design flow control
- Design subjects (e.g. designers, teams, projects)

Design data
describe design objects (e.g. products, systems, etc.), which are the key elements within each design environment. Design data describe the design objects on different **design levels** of abstraction (i.e. levels of design hierarchies) and in different **design domains**.
Models for describing models of design data are called **design object models**. Design object models contain all aspects of a designed product, that are relevant for the design process independent product documentation (functionality, physical properties, geometry, etc.). Since design data depend heavily on the considered industry, the specificatiopn of an overall design object model is not possible. Furthermore, the specification of an industry-wide design object model causes severe problems, because of specific properties of the related design processes and companies, respectively. In some industries, there are standardization efforts for design object models on the way, e.g. EDIF (cf. e.g. [EDIF87]), STEP (cf. e.g. [STEP88]), VHDL (cf. e.g. [VHDL85]), etc.
Modeling languages for describing design object models are traditional database models extended by the well known abstraction concepts, i.e. classification, aggregation, generalization, and association.

Design tools
(e.g. floorplanners, simulators, etc.) constitute the means used by the engineers to progress the design. Design tools are the active components within design environments. For instance, for each design tool, the performed design tasks, the required input data, the expected ouput data, etc. must be described. Furthermore conditions, environment variables (e.g. parameters, options, runtime environment, log protocols, etc.) used for the design tool executions must be described.
All information about design tools are modelled in the **design tool model**. The *CAD Framework Initiative* (CFI) tries to standardize a design tool model as **Tool Encapsulation Specification** (cf. [CFI91]). Here, it is abstracted from the specific properties of design tools

yielding exaclty those information, which are relevant for CAD environments. This concept is called **design tool abstraction.**

Design flow control
is the major component of frameworks. A design flow consists of a sequence of design activities i.e. design tool applications. Since, parallel applications are also possible, design flows can be seen as suitable acyclic graphs, where the nodes represent activities, and the arcs represent the dependencies between activities. Thus, design flows describe the way how a design objecs is designed by the design subjects using the design tools. The terms *design flow* and *design process* are used synonymeously.
Design flow models specify the possible design flows for a certain design methodology. Thus, design flow models allow one to determine correct design flows. Correctness is defined from a methodology point of view.
Modeling languages for design flow models are often petri net like formalisms, where firing of transistions corresponds to activating design tools, and tokens are associated to design objects or, in general, to design situations.
We believe, modeling design flows appropriately requires additionally a rule based description capability for specifying a methodology based control flow. For instance, in several design situations it must be possible to formulate quantified preconditions for the continuation of the design flow (cf. [BD86]).

Design subjects
are involved in design processes, e.g. (1) designers, as users of design tools, (2) tool integrators, who integrate new tools into frameworks, (3) developper of design methodologies, who plan design processes, (4) project managers, who manage design projects (cf. [FKKP90]). These different user classes need different parts of the other information classes (e.g. design object data, design tool data).
Design subject models describe the different user types in design environments and their access and authorization rights. Design subject models are well known in database technologies.

2.2 Design Structure Model

As discussed in the previous sections, the design flow control needs information about the design tools as well as about the design objects. Whereas, the instances of the design tool model represent suitable information about design tools for design flow control, the data of design object models do **not**.
First, the data of design object models are too detailed, i.e. the granularity of design data used by design tools is too small for design flow control. For instance, the application of a certain router rather depends on the existence of a whole netlist, than on the existence of certain pins.
Secondly, some important information are missing, which are relevant for design flow control, e.g. data about versions and alternatives ofdesign objects. For the design tool itself, this information is not relevant, because it always gets exactly one schematic as input data.

Summarizing, a suitable model representing these kinds of information is necessary. This model must represent the following information required by the design flow control:

- Information about the existence of design data in a large granularity. The design flow control must be informed about the status of the design process for activating the correct design tools.
- Information about the design hierarchy.
- Information about the history of design data, e.g. which design tools were activated in which sequence, and which parameter values were used.
- Information about versions, alternatives, etc. (i.e. version model)
- Information about the design structure hierarchy of each design process
- Information about configurations among different design structure hierarchy levels (i.e. configuration model)

The related model is called **design structure model**, because it models information about the structure of design processes. The design structure model abstracts form the design data, such that only the relevant aspects for controlling and monitoring design processes by CAD frameworks are represented.
Whereas design object models are tool specific, the design structure model is tool independent. Hence, the design structure model can be combined with various design object models.

More important, information, which is modelled today in several design object models, but not used by the design tools, need not to be modelled any more by the design object models (e.g. design hierarchy, version model, configuration model). Thus, data inconsistency problems yielding from data redundancies are avoided.

The abstraction concept applied for building the design structure model can be compared with the tool abstraction concept applied for building design tool models. Hence, it is called **object abstraction concept**. Therefore, the entities of an instance of the design structure model are called **meta objects**.

Today, there are strong standardization efforts for **product models** on the way (e.g. EDIF (cf. e.g. [EDIF87]), STEP (cf. e.g. [STEP88]), VHDL (cf. e.g. [VHDL85]), etc.). But, these product models are developed and standardized without differentiating between the issues concerning design object models and the issues related to design structure models. Thus, it is difficult to relate the different product models to a common design structure model, which is used by several frameworks. Furthermore, there are no standardization efforts for design structure models.

2.3 Modeling Design Methodologies

Since the design methodology forms the most important issue in modeling design environments, it must be explicitly described in the design model. Design methodology is often reduced to the aspect of design flow, because design flows give a criterion to distinguish design methodologies. But in our opinion, design methodology is strongly influenced additionally by the available design tools, and by several particular aspects of the design objects, i.e. the design structure.

In figure 3 all five partial models and their relationships and interfaces are shown. A possible refinement of each partial model is illustrated by a simple example.

Relationships of the design object model

The relationship between the design object model and the **design structure model** was already discussed in the previous section. The relationship is determined by the fact that each object model represents a particular aspect of a design object. As shown in figure 3, the relationships between the design object model and the design structure model are called **views**. Views are the large granulates of design data in the design object model, which are related to the elements in the design structure model. Examples for views *netlist*, *schematic*, etc. Views form the smallest access granularity of design objects used to describe a design methodology with respect to the design structure model and the design tool model. Of course, this granularity is **not** defined in general, but it has to be adapted according to the used design tools.

Summarizing, the granularity of views is a tradeoff between supplying too much and unnecessary data and complicating the design management by increasing needlessly the number of views.

Relationships of the design tool model

The design data used and generated by the design tools should be modelled by the design tool model in the same granularity as by the design structure model. This granularity results from the data requests of the design tools. The granularity should be fine enough, such that the data exchange among the design tools can be described. But, the granularity must be as coarse as possible. Otherwise, design management is too complicated.

Summarizing, with respect to the design object model, design data are modelled by the design tool model as views. Thus, the relationship between the design tool model and the **design object model** are views. But, the relationship between the design tool model and the **design structure model** are the entities of the design structure model itself, i.e. the **meta objects**, like versions and variants. For instance, this necessary, when design tools should generate versions or variants of a schematic. Since the information about versions and variants should be only modelled in the design structure model, the information given by the design tool model is not sufficient. Furthermore, information about structure hierarchies should only be also modelled by the design structure model. Thus, for accessing design data on different hierarchy levels by a design tool, the related objects in the design structure model must be used.

Relationships of the design flow model

The design flow must point to the design tools as active components of design processes as well as to the design data as passive components of design process. For instance, in a petri net based flow description, the tokens are instances of these entities, i.e. the tokens are an abstraction of the design data as described by the design structure model, and the transitions represent design tool executions. Again, the granularity of design data are views, i.e. the relationship to the **design object model** are views. The relationship between the design flow model and the **design structure model** are the entities of the design structure model itself, i.e. the meta objects. This is necessary for modeling design flows dealing with variants, versions, hierarchies, etc.

Relationships of the design subject model

The design subject model needs the same granularity of design data as the other partial models. Thus, the relationship of the design subject model and **design object model** are views. Furthermore, for coupling user authorizations to versions, hierarchies, etc. the relationship between the design subject model and the **design structure model** are the meta objects of the design structure model, again.

Summarizing, the discussion of the inter relationships of the various partial models shows the central role of the design structure model. Following this approach, modeling of complex flows and methodologies is possible.

Furthermore, the strong relationships among these models require an integration of the partial models yielding an **integrated design model**.

The key characteristics of the given approach are the five widely orthogonal dimensions for describing design environments as well as the clarity and explicity of the interfaces between these dimensions. Thus, the interchange of partial models is easy. For example, a modification of the hierarchy representation or of the versioning concept must not cause a change of the flow model. Only the representation within the design structure model have to be adapted.

2.4 Design Methodology Specification

In our approach, the design tool model, design flow model, and design structure model specify the design methodology. The **design flow model** is necessary, because a a design methodology is implemented by a sequence of design tool executions. The design tools are the active components of design processes, i.e. the **operators**. The operators must be known to the design methodology, i.e. required input data and

expected output data, design tool functionality, etc. Hence, the design tool model is essential for describing the design methodology. But, the design tool model should be independent of a specific design methodology. That means, the description of the design tools should not depend on the chosen design methodolgy.

Because of its central role, the **design structure model** is also necessary for describing design methodologies. For instance, a hierachical top down or bottom up design can be only described by the design structure model. The design structure model should be as generic as possible, such that as much as possible design methodologies can be described.

Following our approach, design methodologies can be described explicitly using these the different partial models. The **benefits** of an explicit description of the design methodology are:

- Complex design processes can be managed and controlled, i.e. they can be planned, traced, reviewed, and recovered, respectively.
- Design engineers can be supported by special tools for project management, e.g. for selecting the most appropriate design tool from the pool of available design tools.
- The entire design process can be optimized in global way.

The approach can be compared with describing algorithms within a common **programming environment**. The most programming languages like PASCAL, C, or MODULA2 require: (1) the specification of the data structures, (2) the specification of procedural primitives, and (3) the procedural specification of the control flow.
Considering this analogy, the definition of a design methodology can be specified by the following steps:

1. The design structure (cf. specification of data structures) is specified in order to determine the information primitives the methodology is based on.
2. The design tools are introduced (cf. specification of procedural primitives). Thus, the operational primitives used by the design methodology are defined.
3. The design flow is defined (cf. specification of the control flow). The design flow represents the dynamic and strategic aspects of the design methodology.

Summarizing, if these aspects of design objects are not described in the design structure model, they must be modeled either in the design flow model or in the design object model. Both cases yield to an unnatural and redundant modeling. The flow model and object model are semantically overloaded. Moreover, redundant modeling leads to consistency problems. Hence, the structure model allows a non redundant and uniform modeling of design environments in a singe design model. This is a major basis of an efficient Design Methodology Management.

3 Comparision of Related Work

In the following, we would like to discuss the advantages of the proposed design model in more detail. Therefore, we compare our modeling approach with the concepts given by the references. In [BKLHW90] design flows of the VENUS system are described by extended predicate transistion petri nets. The *Hierarchical Specification* contains tokens with the hierarchy relationship of the tokens at the other places, which represent design data of the domain *structure* (e.g. netlists). This kind of flow model does not only describe dynamic aspects, but deals also with static aspects like hierarchies. Thus, the hierarchy information must also be represented in the database of the underlying HILDA framework (cf. [HH87]). Obviously, the management of this redundant information is very disadvantageous from a data modeling point of view. The consistency of multiple representations of the same fact has to be controlled by the CAD framework. For instance, even a simple delete operation on the database requires the deletion of all corresponding hierarchy tokens without disturbing the flow execution. Hence, the delete operation consists of several suboperations, and the delete operation must be handled as a transaction responding to the ACID principle (cf. [Gr81], [HR83]). This means, if this transaction is aborted for some reason, the state before starting the delete operation must be recovered. Hence, all removed tokens must be restored without getting in conflict with the pending transitions and the firing rule of the petri net.

Considering that the hierarchy representation within HILDA meets only simple cases (in reality the design objects have versions, alternatives, and configurations), we can imaging that this flow model becomes very complicated and the control of correct execution nearly seems to be impossible. Summarizing, this kind of flow model is inadequate for modeling hierarchies, versioning, and configurations.
The conclusion of all these problems shows the demand for clearly separating the design flow description and the design structure in two related partial models to avoid redundancy.

4 Conclusion

To meet the needs of design tool developers as well as of design methodology management, a clear separation of the design data model into design structure model and design object model is necessary. These two models decribe the design data on different abstraction levels and granularities as used by the design tool developer and design methodology management, repectively. This seperation and the integrated modelling of the relationships of the design flow model and design tool model to the design structure model enables the interchangeability of design tools, design methodolgies, and product models (i.e. one design structure model for all design object models) within CAD frameworks.

The design structure model restricts and organizes the set of possible design methodologies. For instance, it makes no sense to use design alternatives and libraries, if alternatives or instances of design objects are not represented efficiently by the design structure mo-

del. Thus, one of the first goals of the developers of a design system (if not the major one) should be the development of a general, not less expressive design structure model that supports different design styles and all kind of design objects. Because of even the best model cannot meet all future needs, this model should be easily extensible. Hence, the design structure model should be seperated from the other partial models of the design model. Thus, changes of the design structure model require no or only minimal changes of the design object model as well as the design tool model.

Summarizing, only an adequate abstraction both from the CAD tools (design tool model) and the design objetcs (design structure model) integrated in single design model with clear and simple interfaces among the partial models makes the design methodology appropriately manageable.

The ideas addressed in this paper have already influenced the design and will determine the ongoing implementation of an CAD framework prototype supporting VLSI chip design. Currently, we are working on getting more expieriences in carrying the modeling concepts into and to exploit them in the area of mechanical design environments.

References

[ARF91] W. Allen, D.Rosenthal, K. Fiduk: *The MCC CAD Framework Methodology Management System*, in: Proc. 28th ACM/IEEE DAC, 1991.

[BD86] M. Bushnell, S. Director: *VLSI CAD Tool Integration Using the ULYSSES Environment*, in: Proc. 23rd ACM/IEEE DAC, 1986.

[BKLHW90] F. Bretschneider, C. Kopf, H. Lagger, A. Hsu, E. Wei: *Knowledge Based Design Flow Management*, in: Proc. ICCAD 90, 1990.

[CFI91] K. Fiduk: *Tool Encapsulation Specification*, CAD Framework Initiative, Document No. 51, 1991.

[EDIF87] *EDIF - Electronic Design Interchange Format, Version 2 0 0*, Electronics Industries Association, 1987.

[FKKP90] F.W. Fiduk, S. Kleinfeldt, M. Kosarchyn, E.B. Perez: *Design Methodology Management - A CAD Framework Initiative Perspective -*, in: Proc. 27th ACM/IEEE DAC, 1990.

[Gr81] J. Gray: *The Transaction Concept: Virtues and Limitations*, in: Proc. 7th VLDB, 1981.

[HH87] A. Hsu, L. Hsu: *HILDA: An Integrated System Design Environment*, in: Proc. ICCAD 87, 1987.

[HNSB90] D.S. Harrison, A.R. Newton, R.L. Spickelmier, T.J. Barnes: *Electronic CAD Frameworks*, in: Proc. of the IEEE, Vol. 78, No. 2, 1990.

[HR83] Th. Härder, A. Reuter: *Principles of Transaction-Oriented Database Recovery*, ACM Computing Surveys, Vol. 15, No. 4, 1983.

[Ka85] R.H. Katz: *Information Management for Engineering Design*, Springer, 1985.

[Si91] E. Siepmann: *Do We Need a Common Standard for the Design Structure Model?* in: Proc. 2. IFIP WS on Electronic Design Automation Frameworks, F.J. Ramming and R. Waxman (Ed.), North-Holland, 1991

[SZ89] E. Siepmann, G. Zimmermann: *An Object-Oriented Database Model for the VLSI Design System PLAYOUT*, in: Proc. 26th ACM/IEEE DAC, 1989.

[STEP88] *STEP - Preliminary Design*, ISO TC 184/SC4/WG1 N208, 1988.

[TAS90] CAD Framework Initiative Draft Proposal: *Design Methodology Management - Tool Abstraction Specification*, 1990.

[VHDL85] *The VHDL Language Reference Manual*, Intermetrics Inc., 1985.

[We89] S. Wendt: *Nichtphysikalische Grundlagen der Informationstechnik*, Springer, 1989.

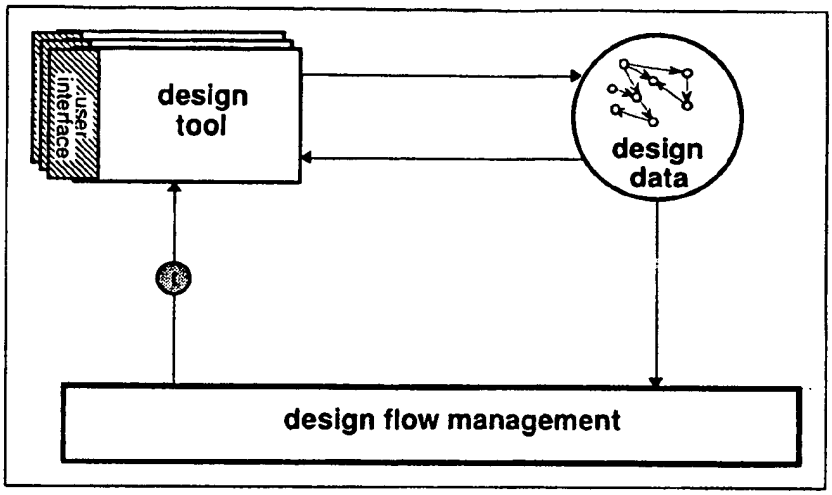

Figure 1: Design Environment for a Design Centered Approach

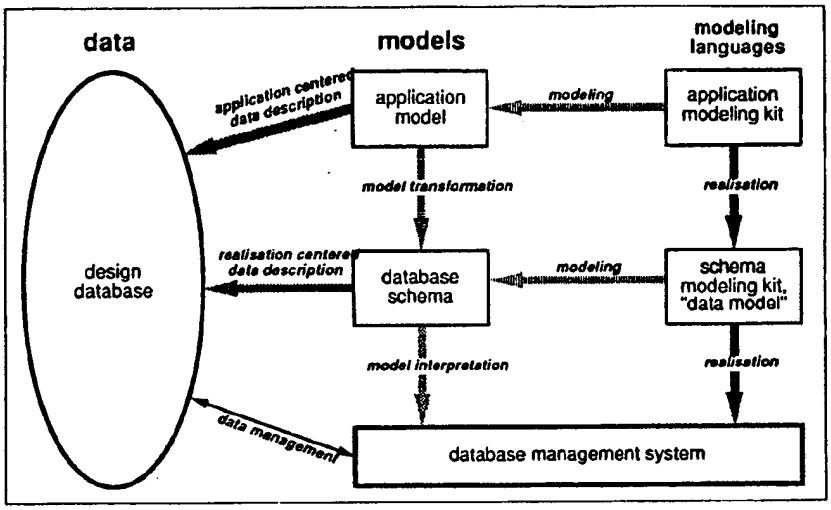

Figure 2: Modeling Levels

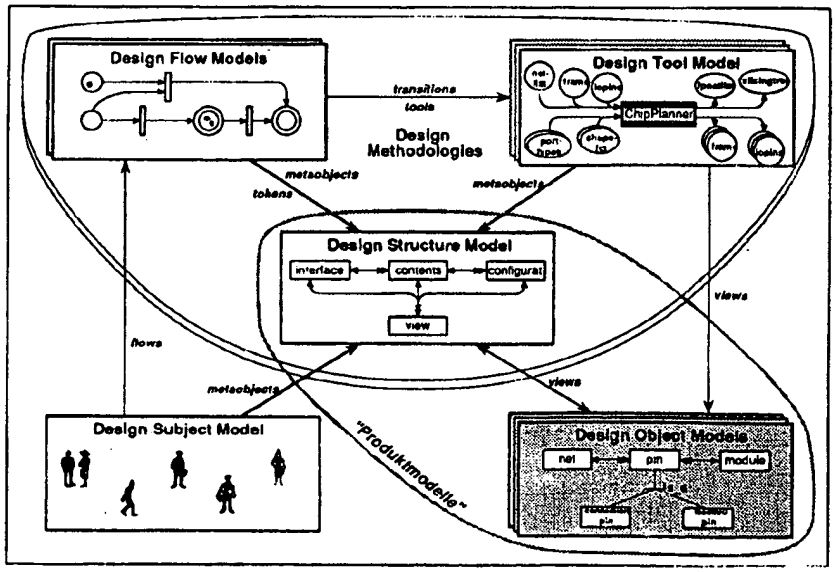

Figure 3: Design Model